1fw AF

**PATENT APPLICATION**

HEWLETT-PACKARD COMPANY
Intellectual Property Administration
P.O. Box 272400
Fort Collins, Colorado 80527-2400

ATTORNEY DOCKET NO. 100110992-1

**IN THE
UNITED STATES PATENT AND TRADEMARK OFFICE**

| | | | |
|---|---|---|---|
| Inventor(s): | John Joseph MAZZITELLI | Confirmation No.: | 1932 |
| Application No.: | 10/057,135 | Examiner: | Serrao, Ranodhi N. |
| Filing Date: | October 29, 2001 | Group Art Unit: | 2141 |

Title: MULTI-THREADED SERVER ACCEPT SYSTEM AND METHOD

Mail Stop Appeal Brief - Patents
Commissioner For Patents
PO Box 1450
Alexandria, VA 22313-1450

# TRANSMITTAL OF REPLY BRIEF

Transmitted herewith is the Reply Brief with respect to the Examiner's Answer mailed on December 17, 2007.

This Reply Brief is being filed pursuant to 37 CFR 1.193(b) within two months of the date of the Examiner's Answer.

(Note: Extensions of time are not allowed under 37 CFR 1.136(a))

(Note: Failure to file a Reply Brief will result in dismissal of the Appeal as to the claims made subject to an expressly stated new ground rejection.)

No fee is required for filing of this Reply Brief.

If any fees are required please charge Deposit Account 08-2025.

☒ I hereby certify that this correspondence is being deposited with the United States Postal Service as first class mail in an envelope addressed to: Commissioner for Patents, Alexandria, VA 22313-1450

Date of Deposit: February 18, 2008

**OR**

☐ I hereby certify that this paper is being transmitted to the Patent and Trademark Office facsimile number (571) 273-8300.
Date of facsimile:

Typed Name: Cindy C. Dioso
Signature: Cindy C. Dioso

Respectfully submitted,
John Joseph MAZZITELLI
By James L. Baudino
James L. Baudino
Attorney/Agent for Applicant(s)

Reg No. : 43,486
Date : February 18, 2008
Telephone : 214-855-7544

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

**APPEAL FROM THE EXAMINER TO THE BOARD OF PATENT APPEALS AND INTERFERENCES**

| | | |
|---|---|---|
| In re Application of: | John Joseph MAZZITELLI | Confirmation No.: 1932 |
| Serial No.: | 10/057,135 | |
| Filing Date: | October 29, 2001 | |
| Group Art Unit: | 2141 | |
| Examiner: | Serrao, Ranodhi N. | |
| Title: | MULTI-THREADED SERVER ACCEPT SYSTEM AND METHOD | |
| Docket No.: | 100110992-1 | |

**MAIL STOP: APPEAL BRIEF-PATENTS**
Commissioner for Patents
P.O. Box 1450
Alexandria, Virginia 22313-1450

Dear Sir:

## REPLY BRIEF

Appellant respectfully submits this Reply Brief in response to the Examiner's Answer mailed December 17, 2007, pursuant to 37 C.F.R. § 1.193(b).

## STATUS OF CLAIMS

Claims 1-6, 8-16, 18-26 and 28-30 stand rejected pursuant to a final Office Action mailed May 17, 2007. Claims 7, 17 and 27 have been cancelled without prejudice or disclaimer in a Response to Office Action filed June 29, 2006. Claims 1-6, 8-16, 18-26 and 28-30 are presented for appeal.

## GROUNDS OF REJECTION TO BE REVIEWED ON APPEAL

1. Claims 1, 2 4-6, 8-11, 12-13, 15-16, 18-23, 25-26, and 28-30 were rejected under 35 U.S.C. §102(e) as being anticipated by U.S. Patent Publication No. 2003/0088609 issued to Guedalia et al. (hereinafter "*Guedalia*").

2. Claims 3, 14, and 24 were rejected under 35 U.S.C. §103(a) as being unpatentable over *Guedalia* as applied to respective independent Claims 1, 12, and 22 and further in view of U.S. Patent Publication No. 2001/0029548 issued to Srikantan et al. (hereinafter "*Srikantan*").

## ARGUMENT

1. 35 U.S.C. § 102 rejections

a. Claims 1, 2, 4-6, 8-11

Claims 1, 2 4-6, and 8-11 were rejected under 35 U.S.C. §102(e) as being anticipated by U.S. Patent Publication No. 2003/0088609 issued to *Guedalia* et al. (hereinafter "*Guedalia*"). Appellant respectfully submits that Claim 1 is patentable over the cited references and, therefore, Claims 2, 4-6, and 8-11 that depend respectively, therefrom, are also patentable.

Independent Claim 1 recites "creating a socket accept thread by a control thread of a server process" and "transferring the request to a client thread dynamically created by the control thread to process request data associated with the request" (emphasis added). Appellant respectfully submits that *Guedalia* does not disclose or even suggest all the limitations of Claim 1. For example, *Guedalia* appears to disclose a server software 114 of *Guedalia* configured to manage accept requests from a client and initiate processing of the requests. (*Guedalia*, paragraph 0255). The server software 114 of *Guedalia* appears to comprise a thread manager 116 of *Guedalia* configured to monitor currently running threads and optimize performance of processing requests using the threads. (*Guedalia*, paragraph 0256). The server software 114 of *Guedalia* appears to perform an accept method to accept requests from a client. (*Guedalia*, paragraph 0257). Once requests are accepted, they are placed in queue until the thread manager 116 matches the request in queue with a thread to process the request. A watchdog thread within the thread manager 116 is used to monitor the threads processing the requests. (*Guedalia*, paragraph 0111). However, neither the server software 114 nor the thread manager 116 appear to be threads themselves and, therefore, neither discloses a "control thread" as recited by Claim 1. Moreover, the threads controlled by the thread manager 116 of *Guedalia*, including the watchdog thread of *Guedalia*, do not appear to disclose or even suggest a "control thread" of Claim 1, because the threads of *Guedalia* appear to be configured to process client requests and the watchdog thread appears to be configured to manage the threads processing the requests. Therefore, Appellants respectfully submit that *Guedalia* does not disclose or even suggest "creating a socket accept thread by a control thread of a server process" and "transferring the request to a client thread dynamically created by the control thread to process request data associated with the request" as recited by Claim 1 (emphasis added). Accordingly, for at least these reasons, Appellant respectfully submit that Claim 1 is patentable over *Guedalia*.

Moreover, in the Examiner's Answer, Appellee appears to assert that Appellant relies on the number of created threads as a basis for Appellant's arguments. (Examiner's Answer dated December 17, 2007, page 6). However, Appellee appears to contend that the number of created threads is an inappropriate basis to rely on because the number of created threads is not a limitation recited by Claim 1. (*Id.*). Appellant respectfully submits that the Appellee has misconstrued Appellant's arguments. For example, Appellant cites to the number of created threads in *Guedalia* to indicate that the total number of threads in *Guedalia* appears to remain constant. (*Guedalia*, paragraph 0112). A new thread in *Guedalia* appears to be created when a thread is removed because a tick count for the removed thread has exceeded a particular threshold. (*Guedalia*, paragraph 0245). Thus, Appellant respectfully contends that the creation of the new thread of *Guedalia* is not "dynamically created ...to process request data associated with the request" as recited by Claim 1 (emphasis added). Instead, the new thread is created because an old thread has been removed. Thus, Appellant is not adding any new matter or limitations into Claim 1, and, therefore, disagrees with Appellee's apparent assertion that Appellant is relying on such. Thus, for at least these reasons, Appellant respectfully submits that Claim 1 is patentable over *Guedalia*.

Furthermore, in the Examiner's Answer, the Appellee appears to assert that Appellant contradictorily argues that *Guedalia* does not disclose or even suggests "transferring the request to a client thread dynamically created by the control thread to process request data associated with the request" as recited by Claim 1. (Examiner's Answer dated December 17, 2007, page 7). Specifically, the Appellee appears to contend that Appellant first indicates that *Guedalia* does not appear to disclose "a client thread dynamically created" as recited by Claim 1, but Appellant then also argues that *Guedalia* creates "a new thread to replace another thread in a thread pool." (*Id.*). Appellant respectfully submits that the Appellee has misunderstood Appellant's argument. For example, *Guedalia* appears to disclose a multi-thread management system for processing requests by a server from a client. (*Guedalia*, Abstract and paragraph 0240). *Guedalia* appears to create a finite number of threads within a thread pool to process the requests. (*Guedalia*, 0244). *Guedalia* appears to use a "watchdog" thread to monitor the activity associated with each thread using a tick counter. (*Guedalia*, 0240 and 0243). In *Guedalia*, at every 50 msec interval, the "watchdog" appears to check to see if a particular thread is active, and, if so, increments the tick counter for that particular thread. (*Guedalia*, 0244). If the tick counter for a particular thread in *Guedalia* exceeds a particular threshold (e.g., 3), then the "watchdog" appears to lower the priority of the thread, remove the thread from the thread pool, and create a new thread to replace that thread in the thread pool. (*Guedalia*, paragraph 0245). Thus, *Guedalia* appears to create a new thread based on whether the tick counter of *Guedalia* exceeds particular threshold and does not appear to be "dynamically created...to process request data associated with the request" as recited by Claim 1 (emphasis added). Therefore, for at least these reasons, Appellant respectfully submits that *Guedalia* does not disclose or even suggest all the limitations of Claim 1.

In the Examiner's Answer, the Appellee appears to further argue that *Guedalia* does in fact disclose the limitation of "a client thread dynamically created." (Examiner's Answer dated December 17, 2007, page 7). Specifically, the Appellee references paragraph 0118 of *Guedalia* which recites:

> The present invention describes a dynamic functionality for thread management, in which additional threads are created and destroyed during processing, depending on how busy the server gets.

(*Guedalia*, paragraph 0118) (emphasis added). The cited text of *Guedalia* appears to disclose dynamic functionality associated with "thread management," which relates to the above-mentioned method of creating and destroying threads based on a tick counter for a particular thread exceeding a particular threshold. Therefore, as indicated above, the creation of the new thread of *Guedalia* is not "dynamically created...to process request data associated with [a] request" as recited by Claim 1 (emphasis added). In fact, Appellant respectfully submits that *Guedalia* appears to teach away from the aforementioned limitation recited by Claim 1. For example, *Guedalia* states:

> When a request queues up, the system of the present invention does not immediately create a new thread. Rather, the watchdog manages the threads. Whenever the watchdog discovers, during its regular check, that the tick counter of a thread has reached 3, it then lowers the priority of this thread and removes it from the thread pool, and creates a new thread to replace it. The old thread that was removed from the thread pool completes its task and dies.

(*Guedalia*, paragraph 0245). *Guedalia* further recites:

> The curve labeled 104 in FIG. 3 corresponds to a server using a preferred embodiment of the present invention, and is approximately four times faster than the IIS server. This is because the present invention dynamically allocates threads using a "watchdog" algorithm to monitor threads rather than requests, and was able to process all of the client requests with only 2-3 threads.

(*Guedalia*, paragraph 0252). Additionally, *Guedalia* recites:

> The ITS server allocated approximately 54 threads, and assigned threads to each client. These threads competed for memory pages, and as a result page faults were rampant.

(*Guedalia*, paragraph 0253). *Guedalia* also recites:

> In contrast, the server of the present invention, whose performance is indicated in curve 104 waited a short time (approximately 150 msec.) before assigning threads to client requests, to see if an active thread in the thread pool would be freed up and could then be re-used for processing a queued request. Even though the client requests had to wait in a queue, the overall performance was better due to the fact that there were a smaller number of concurrent active threads. Using the IIS server, client requests did not necessarily have to wait in a queue, and were immediately assigned to a waiting thread, but the proliferation of threads caused enough page faults that the overall performance was worse.

(*Guedalia*, paragraph 0254). The cited text of *Guedalia* appears to indicate that *Guedalia* "waits" before any new threads are created in order to prevent the "proliferation of threads" which could result in additional page faults and affect the performance of the system. (*Id.*). *Guedalia* further appears to disclose that threads are created when a thread is removed from the thread pool, and when created, is placed into the thread pool. Thus, because *Guedalia* "waits" before any new threads are created until the tick counter for a thread exceeds a threshold and is, therefore, removed, *Guedalia* appears to process request data from already existing threads in a thread pool. Therefore, Appellant respectfully submits that *Guedalia* appears to teach away from "transferring the request to a client thread dynamically created by the control thread to process request data associated with the request" as recited by Claim 1 (emphasis added). Accordingly, for at least these reasons, Appellant respectfully submits that *Guedalia* does not appear to teach or suggest all the limitations of Claim 1.

Moreover, in the Examiner's Answer, the Appellee appears to assert that the "waiting" for the creation of a new thread "has no effect on the claim language." (Examiner's Answer dated December 17, 2007, page 8). The Appellee further appears to assert that Claim 1 is "reasonably interpreted to mean that threads are created as requests are received." Therefore, since *Guedalia* appears to indicate that a "new thread, immediately upon its create, is free to process a request in the queue if a queued request exists," Appellee appears to contend that *Guedalia* discloses "transferring the request to a client thread dynamically created by the control thread to process request data associated with the request" as recited by Claim 1. Appellant disagrees. For example, as shown above, *Guedalia* appears to create a new thread in response to the tick count for an existing thread exceeding a particular threshold and the existing thread being removed from the thread pool. Regardless of whether the new thread is free to process data in a queue, the new thread is not "dynamically created... to process request data" as recited by Claim 1 (emphasis added). Instead, the new thread is generated in response to the tick count exceeding a particular threshold. Accordingly, for at least this reason, Appellant respectfully submits that Claim 1 is patentable over *Guedalia*.

Claims 2-6, 8-11, 13-16, 18-21, 23-26 and 28-30 that depend respectively from independent Claim 1 are, therefore, also patentable. Thus, Appellant respectfully submits that the rejection of Claims 1, 2, and 4-11 is improper. Accordingly, Appellant respectfully requests that Claims 1, 2, 4-6, and 8-11 be allowed.

b. Claims 12-13, 15-16, and 18-21

Claims 12-13, 15-16, and 18-21 were rejected under 35 USC §102(e) as being anticipated by *Guedalia*. Appellants respectfully submit that independent Claim 12 is patentable over the cited reference and, thus, remaining Claims 13, 15-16, and 18-21, which depend from independent Claim 12, is also patentable.

Independent Claim 12 recites "a server process residing on a server and operation to create a socket accept thread by a control thread of a server process residing on the server" and "transfer the request to a client thread dynamically created by the control thread to process request data associated with the request" (emphasis added). Appellant respectfully submits that *Guedalia* does not disclose or even suggest all the limitations of Claim 12. For example, *Guedalia* appears to disclose a server software 114 of *Guedalia* configured to manage accept requests from a client and initiate processing of the requests. (*Guedalia*, paragraph 0255). The server software 114 of *Guedalia* appears to comprise a thread manager 116 of *Guedalia* configured to monitor currently running threads and optimize performance of processing requests using the threads. (*Guedalia*, paragraph 0256). The server software 114 of *Guedalia* appears to perform an accept method to accept requests from a client. (*Guedalia*, paragraph 0257). Once requests are accepted, they are placed in queue until the thread manager 116 matches the request in queue with a thread to process the request. A watchdog thread within the thread manager 116 is used to monitor the threads processing the requests. (*Guedalia*, paragraph 0111). However, neither the server software 114 nor the thread manager 116 appear to be threads themselves and, therefore, neither discloses a "control thread" as recited by Claim 12. Moreover, the threads controlled by the thread manager 116 of *Guedalia*, including the watchdog thread of *Guedalia*, do not appear to disclose or even suggest a "control thread" of Claim 12, because the threads of *Guedalia* appear to be configured to process client requests and the watchdog thread appears to be configured to manage the threads processing the requests. Therefore, Appellants respectfully submit that *Guedalia* does not disclose or even suggest "creat[ing] a socket accept thread by a control thread of a server process" and "transfer[ring] the request to a client thread dynamically created by the control thread to process request data associated with the request" as recited by Claim 12 (emphasis added). Accordingly, for at least these reasons, Appellant respectfully submit that Claim 12 is patentable over *Guedalia*.

Moreover, in the Examiner's Answer, Appellee appears to assert that Appellant relies on the number of created threads as a basis for Appellant's arguments. (Examiner's Answer dated December 17, 2007, page 6). However, Appellee appears to contend that the number of created threads is an inappropriate basis to rely on because the number of created threads is not a limitation recited by Claim 12. (*Id.*). Appellant respectfully submits that the Appellee has misconstrued Appellant's arguments. For example, Appellant cites to the number of created threads in *Guedalia* to indicate that the total number of threads in *Guedalia* appears to remain constant. (*Guedalia*, paragraph 0112). A new thread in *Guedalia* appears to be created when a thread is removed because a tick count for the removed thread has exceeded a particular threshold. (*Guedalia*, paragraph 0245). Thus, Appellant respectfully contends that the creation of the new thread of *Guedalia* is not "dynamically created ...to process request data associated with the request" as recited by Claim 12 (emphasis added). Instead, the new thread is created because an old thread has been removed. Thus, Appellant is not adding any new matter or limitations into Claim 12, and, therefore, disagrees with Appellee's apparent assertion that Appellant is relying on such. Thus, for at least these reasons, Appellant respectfully submits that Claim 12 is patentable over *Guedalia*.

Furthermore, in the Examiner's Answer, the Appellee appears to assert that Appellant contradictorily argues that *Guedalia* does not disclose or even suggests "transferring the request to a client thread dynamically created by the control thread to process request data associated with the request" as recited by Claim 12. (Examiner's Answer dated December 17, 2007, page 7). Specifically, the Appellee appears to contend that Appellant first indicates that *Guedalia* does not appear to disclose "a client thread dynamically created" as recited by Claim

12, but Appellant then also argues that *Guedalia* creates "a new thread to replace another thread in a thread pool." (*Id.*). Appellant respectfully submits that the Appellee has misunderstood Appellant's argument. For example, *Guedalia* appears to disclose a multi-thread management system for processing requests by a server from a client. (*Guedalia*, Abstract and paragraph 0240). *Guedalia* appears to create a finite number of threads within a thread pool to process the requests. (*Guedalia*, 0244). *Guedalia* appears to use a "watchdog" thread to monitor the activity associated with each thread using a tick counter. (*Guedalia*, 0240 and 0243). In *Guedalia*, at every 50 msec interval, the "watchdog" appears to check to see if a particular thread is active, and, if so, increments the tick counter for that particular thread. (*Guedalia*, 0244). If the tick counter for a particular thread in *Guedalia* exceeds a particular threshold (e.g., 3), then the "watchdog" appears to lower the priority of the thread, remove the thread from the thread pool, and create a new thread to replace that thread in the thread pool. (*Guedalia*, paragraph 0245). Thus, *Guedalia* appears to create a new thread based on whether the tick counter of *Guedalia* exceeds particular threshold and does not appear to be "dynamically created...to process request data associated with the request" as recited by Claim 12 (emphasis added). Therefore, for at least these reasons, Appellant respectfully submits that *Guedalia* does not disclose or even suggest all the limitations of Claim 12.

In the Examiner's Answer, the Appellee appears to further argue that *Guedalia* does in fact disclose the limitation of "a client thread dynamically created." (Examiner's Answer dated December 17, 2007, page 7). Specifically, the Appellee references paragraph 0118 of *Guedalia* which recites:

> The present invention describes a dynamic functionality for thread management, in which additional threads are created and destroyed during processing, depending on how busy the server gets.

(*Guedalia*, paragraph 0118) (emphasis added). The cited text of *Guedalia* appears to disclose dynamic functionality associated with "thread management," which relates to the above-mentioned method of creating and destroying threads based on a tick counter for a particular thread exceeding a particular threshold. Therefore, as indicated above, the creation of the new thread of *Guedalia* is not "dynamically created...to process request data associated with [a] request" as recited by Claim 12 (emphasis added). In fact, Appellant respectfully submits that *Guedalia* appears to teach away from the aforementioned limitation recited by Claim 12. For example, *Guedalia* states:

> When a request queues up, the system of the present invention does not immediately create a new thread. Rather, the watchdog manages the threads. Whenever the watchdog discovers, during its regular check, that the tick counter of a thread has reached 3, it then lowers the priority of this thread and removes it from the thread pool, and creates a new thread to replace it. The old thread that was removed from the thread pool completes its task and dies.

(*Guedalia*, paragraph 0245). *Guedalia* further recites:

> The curve labeled 104 in FIG. 3 corresponds to a server using a preferred embodiment of the present invention, and is approximately four times faster than the IIS server. This is because the present invention dynamically allocates threads using a "watchdog" algorithm to monitor threads rather than requests, and was able to process all of the client requests with only 2-3 threads.

(*Guedalia*, paragraph 0252). Additionally, *Guedalia* recites:

> The ITS server allocated approximately 54 threads, and assigned threads to each client. These threads competed for memory pages, and as a result page faults were rampant.

(*Guedalia*, paragraph 0253). *Guedalia* also recites:

> In contrast, the server of the present invention, whose performance is indicated in curve 104 waited a short time (approximately 150 msec.) before assigning threads to client requests, to see if an active thread in the thread pool would be freed up and could then be re-used for processing a queued request. Even though the client requests had to wait in a queue, the overall performance was better due to the fact that there were a smaller number of concurrent active threads. Using the IIS server, client requests did not necessarily have to wait in a queue, and were immediately assigned to a waiting thread, but the proliferation of threads caused enough page faults that the overall performance was worse.

(*Guedalia*, paragraph 0254). The cited text of *Guedalia* appears to indicate that *Guedalia* "waits" before any new threads are created in order to prevent the "proliferation of threads" which could result in additional page faults and affect the performance of the system. (*Id.*). *Guedalia* further appears to disclose that threads are created when a thread is removed from the thread pool, and when created, is placed into the thread pool. Thus, because *Guedalia* "waits" before any new threads are created until the tick counter for a thread exceeds a threshold and is, therefore, removed, *Guedalia* appears to process request data from already existing threads in a thread pool. Therefore, Appellant respectfully submits that *Guedalia* appears to teach away from "transferring the request to a client thread dynamically created by the control thread to process request data associated with the request" as recited by Claim 12 (emphasis added). Accordingly, for at least these reasons, Appellant respectfully submits that *Guedalia* does not appear to teach or suggest all the limitations of Claim 12.

Moreover, in the Examiner's Answer, the Appellee appears to assert that the "waiting" for the creation of a new thread "has no effect on the claim language." (Examiner's Answer dated December 17, 2007, page 8). The Appellee further appears to assert that Claim 12 is "reasonably interpreted to mean that threads are created as requests are received." Therefore, since *Guedalia* appears to indicate that a "new thread, immediately upon its create, is free to process a request in the queue if a queued request exists," Appellee appears to contend that *Guedalia* discloses "transferring the request to a client thread dynamically created by the control thread to process request data associated with the request" as recited by Claim 12. Appellant disagrees. For example, as shown above, *Guedalia* appears to create a new thread in response to the tick count for an existing thread exceeding a particular threshold and the existing thread being removed from the thread pool. Regardless of whether the new thread is free to process data in a queue, the new thread is not "dynamically created... to process request data" as recited by Claim 12 (emphasis added). Instead, the new thread is generated in response to the tick count exceeding a particular threshold. Accordingly, for at least this reason, Appellant respectfully submits that Claim 12 is patentable over *Guedalia.*

Claims 13, 15-16, and 18-21 that depend respectively from independent Claim 12 are, therefore, also patentable. Thus, Appellant respectfully submits that the rejection of Claims 12-13, 15-16, and 18-21 is improper. Accordingly, Appellant respectfully requests that Claims 12-13, 15-16, and 18-21 be allowed.

c. Claims 22-23, 25-26, and 28-30

Claims 22-23, 25-26, and 28-30 were rejected under 35 USC §102(e) as being anticipated by *Guedalia.* Appellants respectfully submit that independent Claim 22 is patentable over the cited reference and, thus, remaining Claims 23, 25-26, and 28-30, which depend from independent Claim 22, is also patentable.

Independent Claim 22 recites "an application software residing on a computer-readable medium and operable to: create a socket accept thread by a control thread of the application software" and "transfer the request to a client thread dynamically created by the control thread to process request data associated with the request" (emphasis added). Appellant respectfully submits that *Guedalia* does not disclose or even suggest all the limitations of Claim 22. For example, *Guedalia* appears to disclose a server software 114 of *Guedalia* configured to manage accept requests from a client and initiate processing of the requests. (*Guedalia*, paragraph 0255). The server software 114 of *Guedalia* appears to comprise a thread manager 116 of *Guedalia* configured to monitor currently running threads and optimize performance of processing requests using the threads. (*Guedalia*, paragraph 0256). The server software 114 of *Guedalia* appears to perform an accept method to accept requests from a client. (*Guedalia*, paragraph 0257). Once requests are accepted, they are placed in queue until the thread manager 116 matches the request in queue with a thread to process the request. A watchdog thread within the thread manager 116 is used to monitor the threads processing the requests. (*Guedalia*, paragraph 0111). However, neither the server software 114 nor the thread manager 116 appear to be threads themselves and, therefore, neither discloses a "control thread" as recited by Claim 22. Moreover, the threads controlled by the thread manager 116 of *Guedalia*, including the watchdog thread of *Guedalia*, do not appear to disclose or even suggest a "control thread" of Claim 22, because the threads of *Guedalia* appear to be configured to process client requests and the watchdog thread appears to be configured to manage the threads processing the requests. Therefore, Appellants respectfully submit that *Guedalia* does not disclose or even suggest "creat[ing] a socket accept thread by a control thread of a server process" and "transfer[ring] the request to a client thread dynamically created by the control thread to process request data associated with the request" as recited by Claim 22 (emphasis added). Accordingly, for at least these reasons, Appellant respectfully submit that Claim 22 is patentable over *Guedalia*.

Moreover, in the Examiner's Answer, Appellee appears to assert that Appellant relies on the number of created threads as a basis for Appellant's arguments. (Examiner's Answer dated December 17, 2007, page 6). However, Appellee appears to contend that the number of created threads is an inappropriate basis to rely on because the number of created threads is not a limitation recited by Claim 22. (*Id.*). Appellant respectfully submits that the Appellee has misconstrued Appellant's arguments. For example, Appellant cites to the number of created threads in *Guedalia* to indicate that the total number of threads in *Guedalia* appears to remain constant. (*Guedalia*, paragraph 0112). A new thread in *Guedalia* appears to be created when a thread is removed because a tick count for the removed thread has exceeded a particular threshold. (*Guedalia*, paragraph 0245). Thus, Appellant respectfully contends that the creation of the new thread of *Guedalia* is not "dynamically created ...to process request data associated with the request" as recited by Claim 22 (emphasis added). Instead, the new thread is created because an old thread has been removed. Thus, Appellant is not adding any new matter or limitations into Claim 22, and, therefore, disagrees with Appellee's apparent assertion that Appellant is relying on such. Thus, for at least these reasons, Appellant respectfully submits that Claim 22 is patentable over *Guedalia*.

Furthermore, in the Examiner's Answer, the Appellee appears to assert that Appellant contradictorily argues that *Guedalia* does not disclose or even suggests "transferring the request to a client thread dynamically created by the control thread to process request data associated with the request" as recited by Claim 22. (Examiner's Answer dated December 17, 2007, page 7). Specifically, the Appellee appears to contend that Appellant first indicates that *Guedalia* does not appear to disclose "a client thread dynamically created" as recited by Claim 22, but Appellant then also argues that *Guedalia* creates "a new thread to replace another thread in a thread pool." (*Id.*). Appellant respectfully submits that the Appellee has misunderstood Appellant's argument. For example, *Guedalia* appears to disclose a multi-thread management system for processing requests by a server from a client. (*Guedalia*, Abstract and paragraph 0240). *Guedalia* appears to create a finite number of threads within a thread pool to process the requests. (*Guedalia*, 0244). *Guedalia* appears to use a "watchdog" thread to monitor the activity associated with each thread using a tick counter. (*Guedalia*, 0240 and 0243). In *Guedalia*, at every 50 msec interval, the "watchdog" appears to check to see if a particular thread is active, and, if so, increments the tick counter for that particular thread. (*Guedalia*, 0244). If the tick counter for a particular thread in *Guedalia* exceeds a particular threshold (e.g., 3), then the "watchdog" appears to lower the priority of the thread, remove the thread from the thread pool, and create a new thread to replace that thread in the thread pool. (*Guedalia*, paragraph 0245). Thus, *Guedalia* appears to create a new thread based on whether the tick counter of *Guedalia* exceeds particular threshold and does not appear to be "dynamically created...to process request data associated with the request" as recited by Claim 22 (emphasis added). Therefore, for at least these reasons, Appellant respectfully submits that *Guedalia* does not disclose or even suggest all the limitations of Claim 22.

In the Examiner's Answer, the Appellee appears to further argue that *Guedalia* does in fact disclose the limitation of "a client thread dynamically created." (Examiner's Answer dated December 17, 2007, page 7). Specifically, the Appellee references paragraph 0118 of *Guedalia* which recites:

> The present invention describes a dynamic functionality for thread management, in which additional threads are created and destroyed during processing, depending on how busy the server gets.

(*Guedalia*, paragraph 0118) (emphasis added). The cited text of *Guedalia* appears to disclose dynamic functionality associated with "thread management," which relates to the above-mentioned method of creating and destroying threads based on a tick counter for a particular thread exceeding a particular threshold. Therefore, as indicated above, the creation of the new thread of *Guedalia* is not "dynamically created...to process request data associated with [a] request" as recited by Claim 22 (emphasis added). In fact, Appellant respectfully submits that *Guedalia* appears to teach away from the aforementioned limitation recited by Claim 22. For example, *Guedalia* states:

> When a request queues up, the system of the present invention does not immediately create a new thread. Rather, the watchdog manages the threads. Whenever the watchdog discovers, during its regular check, that the tick counter of a thread has reached 3, it then lowers the priority of this thread and removes it from the thread pool, and creates a new thread to replace it. The old thread that was removed from the thread pool completes its task and dies.

(*Guedalia*, paragraph 0245). *Guedalia* further recites:

> The curve labeled 104 in FIG. 3 corresponds to a server using a preferred embodiment of the present invention, and is approximately four times faster than the IIS server. This is because the present invention dynamically allocates threads using a "watchdog" algorithm to monitor threads rather than requests, and was able to process all of the client requests with only 2-3 threads.

(*Guedalia*, paragraph 0252). Additionally, *Guedalia* recites:

> The ITS server allocated approximately 54 threads, and assigned threads to each client. These threads competed for memory pages, and as a result page faults were rampant.

(*Guedalia*, paragraph 0253). *Guedalia* also recites:

> In contrast, the server of the present invention, whose performance is indicated in curve 104 waited a short time (approximately 150 msec.) before assigning threads to client requests, to see if an active thread in the thread pool would be freed up and could then be re-used for processing a queued request. Even though the client requests had to wait in a queue, the overall performance was better due to the fact that there were a smaller number of concurrent active threads. Using the IIS server, client requests did not necessarily have to wait in a queue,

> and were immediately assigned to a waiting thread, but the proliferation of threads caused enough page faults that the overall performance was worse.

(*Guedalia*, paragraph 0254). The cited text of *Guedalia* appears to indicate that *Guedalia* "waits" before any new threads are created in order to prevent the "proliferation of threads" which could result in additional page faults and affect the performance of the system. (*Id.*). *Guedalia* further appears to disclose that threads are created when a thread is removed from the thread pool, and when created, is placed into the thread pool. Thus, because *Guedalia* "waits" before any new threads are created until the tick counter for a thread exceeds a threshold and is, therefore, removed, *Guedalia* appears to process request data from already existing threads in a thread pool. Therefore, Appellant respectfully submits that *Guedalia* appears to teach away from "transferring the request to a client thread dynamically created by the control thread to process request data associated with the request" as recited by Claim 22 (emphasis added). Accordingly, for at least these reasons, Appellant respectfully submits that *Guedalia* does not appear to teach or suggest all the limitations of Claim 22.

Moreover, in the Examiner's Answer, the Appellee appears to assert that the "waiting" for the creation of a new thread "has no effect on the claim language." (Examiner's Answer dated December 17, 2007, page 8). The Appellee further appears to assert that Claim 22 is "reasonably interpreted to mean that threads are created as requests are received." Therefore, since *Guedalia* appears to indicate that a "new thread, immediately upon its create, is free to process a request in the queue if a queued request exists," Appellee appears to contend that *Guedalia* discloses "transferring the request to a client thread dynamically created by the control thread to process request data associated with the request" as recited by Claim 22. Appellant disagrees. For example, as shown above, *Guedalia* appears to create a new thread in response to the tick count for an existing thread exceeding a particular threshold and the existing thread being removed from the thread pool. Regardless of whether the new thread is free to process data in a queue, the new thread is not "dynamically created... to process request data" as recited by Claim 22 (emphasis added). Instead, the new thread is generated in response to the tick count exceeding a particular threshold. Accordingly, for at least this reason, Appellant respectfully submits that Claim 22 is patentable over *Guedalia*.

Claims 23, 25-26, and 28-30 that depend respectively from independent Claim 22 are, therefore, also patentable. Thus, Appellant respectfully submits that the rejection of Claims 22-

23, 25-26, and 28-30 is improper. Accordingly, Appellant respectfully requests that Claims 22-23, 25-26, and 28-30 be allowed.

2. <u>35 U.S.C. § 103 rejections</u>

a. <u>Claims 3, 14, and 24</u>

Claims 3, 14, and 24 were rejected under 35 U.S.C. §103(a) as being unpatentable over *Guedalia* as applied to respective independent Claims 1, 12, and 22 and further in view of U.S. Patent Publication no. 2001/0029548 issued to Srikantan et al. (hereinafter "*Srikantan*"). Appellants respectfully submit that Claim 3, 14, and 24 are patentable over *Guedalia* in view of *Srikantan* and are therefore allowable.

Claims 3, 14, and 24 depend from respective independent Claims 1, 12 and 22. Appellant repeats and incorporates herein the arguments presented above in connection with independent Claims 1, 14, and 24 such that *Guedalia* does not disclose or even suggest all the limitations of Claims 1, 14, and 24 and, therefore, *Guedalia* does not disclose or even suggest all the limitations of Claim 3, 14, and 24 which depend from respective Claims 1, 14, and 24. Further, the Appellee does not rely on *Srikantan* to remedy, nor does *Srikantan* appear to remedy, at least the deficiencies of *Guedalia* indicated above. Therefore, for at least this reasons, Claims 3, 14, and 24 are patentable over *Guedalia* in view of *Srinkantan*.

## CONCLUSION

Appellant has demonstrated that the present invention as claimed is clearly distinguishable over the art cited of record. Therefore, Appellant respectfully requests the Board of Patent Appeals and Interferences to reverse the final rejection of the Examiner and instruct the Examiner to issue a notice of allowance of all claims.

No fee is believed due with this Reply Brief. If, however, Appellant has overlooked the need for any fee, the Commissioner is hereby authorized to charge any fees or credit any overpayments to Deposit Account No. 08-2025 of Hewlett-Packard Company.

Respectfully submitted,

James L Baudino

James L. Baudino
Registration No. 43,486

Date: February 18, 2008

Correspondence To:

Hewlett-Packard Company
Intellectual Property Administration
Grenoble, France
Tel. 33 4 76 14 17 98